Distribution-Free Property Testing

Shirley Halevy[1] and Eyal Kushilevitz[2]

[1] Department of Computer Science, Technion, Haifa 3200, Israel.
shirleyh@cs.technion.ac.il.
[2] Department of Computer Science, Technion, Haifa 3200, Israel.
eyalk@cs.technion.ac.il.

Abstract. We consider the problem of distribution-free property testing of functions. In this setting of property testing, the distance between functions is measured with respect to a *fixed but unknown* distribution D on the domain, and the testing algorithms have an oracle access to random sampling from the domain according to this distribution D. This notion of distribution-free testing was previously defined, but no distribution-free property testing algorithm was known for any (non-trivial) property. By extending known results (from "standard", uniform distribution property testing), we present the first such distribution-free algorithms for two of the central problems in this field:

- A distribution-free testing algorithm for low-degree multivariate polynomials with query complexity $O(d^2 + d \cdot \epsilon^{-1})$, where d is the total degree of the polynomial.
- A distribution-free monotonicity testing algorithm for functions $f : [n]^d \to A$ for low-dimensions (e.g., when d is a constant) with query complexity $O(\frac{\log^d n \cdot 2^d}{\epsilon})$.

The same approach that is taken for the distribution-free testing of low-degree polynomials is shown to apply also to several other problems.

1 Introduction

The classical notion of *decision problems* requires an algorithm to distinguish objects having some property $\mathcal{P}$ from those objects which do not have the property. *Property testing* is a recently-introduced relaxation of decision problems, where algorithms are only required to distinguish objects having the property $\mathcal{P}$ from those which are at least "ϵ-far" from every such object. The notion of property testing was introduced by Rubinfeld and Sudan [35] and since then attracted a considerable amount of attention. Property testing algorithms (or *property testers*) were introduced for problems in graph theory (e.g. [2, 23, 24, 30]), monotonicity testing (e.g. [9, 13, 14, 18, 19, 22]) and other properties (e.g. [1, 3–5, 7, 10, 12, 15, 17, 20, 27–29, 31, 32, 34]; the reader is referred to excellent surveys by Ron [33], Goldreich [21], and Fischer [16] for a presentation of some of this work, including some connections between property testing and other topics). The main goal of property testers is to avoid "reading" the whole object (which requires complexity at least linear in the size of its representation); i.e., to make the decision by reading a small (possibly, selected at random) fraction

of the input (e.g., a fraction of size polynomial in $1/\epsilon$ and poly-logarithmic in the size of the representation) and still having a good (say, at least 2/3) probability of success.

A crucial component in the definition of property testing is that of the *distance* between two objects. For the purpose of this definition, it is common to think of objects as being *functions* over some domain $\mathcal{X}$. For example, a graph G may be thought of as a function $f_G : V \times V \to \{0,1\}$ indicating for each edge e whether it exists in the graph. The distance between functions f and g is then measured by considering the set $\mathcal{X}_{f \neq g}$ of all points x where $f(x) \neq g(x)$ and comparing the size of this set $\mathcal{X}_{f \neq g}$ to that of $\mathcal{X}$; equivalently, one may introduce a uniform distribution over $\mathcal{X}$ and measure the probability of picking $x \in \mathcal{X}_{f \neq g}$. Note that property testers access the input function (object) via *membership queries* (i.e., the algorithm gives a value x and gets $f(x)$).

It is natural to generalize the above definition of distance between two functions, to deal with arbitrary probability distributions D over $\mathcal{X}$, by measuring the probability of $\mathcal{X}_{f \neq g}$ according to D. Ideally, one would hope to get *distribution-free* property testers. A distribution-free tester for a given property $\mathcal{P}$ accesses the function using membership queries, as above, and by randomly sampling the **fixed but unknown** distribution D (this mimics similar definitions from *learning theory* and is implemented via an oracle access to D; see, e.g., [26] [1]). As before, the tester is required to accept the given function f with probability at least $\frac{2}{3}$ if f satisfies the property $\mathcal{P}$, and to reject it with probability at least $\frac{2}{3}$ if f is at least ϵ-far from $\mathcal{P}$ with respect to the distribution D.

Indeed, these definitions of distance with respect to an arbitrary distribution D and of distribution-free testing were already considered in the context of property testing [23]. However, to the best of our knowledge, no distribution-free property tester was known for any (non-trivial) property (besides testing algorithms that follow from the existence of proper learning algorithms in learning-theory [23]). Moreover, discouraging impossibility results, due to [23], show that for many graph-theoretic properties (for which testers that work with respect to the uniform distribution are known) no such (efficient) distribution-free tester exists. As a result, most previous work focused on testers for the uniform distribution; some of these algorithms can be generalized to deal with certain (quite limited) classes of distributions (e.g., product distributions [23]), and very few can be modified to be testers with respect to any **known** distribution (as was observed by [16] regarding the tester presented in [28]), but none is shown to be a distribution-free tester. Let us review some of the central problems, studied in the context of property testing, which are relevant to the current work.

Low-degree tests for polynomials. The first problem studied in the field of property testing was that of low-degree testing for multivariate polynomials over a

[1] More precisely, distribution-free property testing is the analogue of the PAC+MQ model of learning (that was studied by the learning-theory community mainly via the EQ+MQ model); standard property testing is the analogue of the uniform+MQ model.

finite field, where one wishes to test whether a given function can be represented by a multivariate polynomial of total degree d, or is it ϵ-far from any such polynomial. Later, the problem of low-degree testing played a central role in the development of probabilistic checkable proofs (PCP), where the goal is to probabilistically verify the validity of a given proof. For the problem of low-degree testing, Rubinfeld and Sudan [35] presented a tester with query complexity of $O(d^2 + d \cdot \epsilon^{-1})$. This test was further analyzed in [8]. The reader is also referred to [10], where a linearity test (which tests whether a given function acts as an homomorphism between groups) is presented, and to [3, 6, 7, 20] for other related work.

Monotonicity testing. *Monotonicity* has also been a subject of a significant amount of work in the property testing literature (e.g. [9, 13–15, 18, 19, 22]). In monotonicity testing, the domain $\mathcal{X}$ is usually the d-dimensional cube $[n]^d$. A partial order is defined on this domain in the natural way (for $\boldsymbol{y}, \boldsymbol{z} \in [n]^d$, we say that $\boldsymbol{y} \leq \boldsymbol{z}$ if each coordinate of $\boldsymbol{y}$ is bounded by the corresponding coordinate of $\boldsymbol{z}$).[2] A function f over the domain $[n]^d$ is *monotone* if whenever $\boldsymbol{z} \geq \boldsymbol{y}$ then $f(\boldsymbol{z}) \geq f(\boldsymbol{y})$. Testers were developed to deal with both the low-dimensional and the high-dimensional cases (with respect to the uniform distribution over the domain). In what follows, we survey some of the known results on this problem. In the low-dimensional case, d is considered to be small compared to n (and, in fact, it is typically a constant); a successful algorithm for this case is typically one that is polynomial in $1/\epsilon$ and in $\log n$. The first paper to deal with this case is by Ergün et al. [14] which presented an $O(\frac{\log n}{\epsilon})$ algorithm for the line (i.e., the case $d = 1$), and showed that this query complexity cannot be achieved without using membership queries. This algorithm was generalized for any fixed d in [9]. For the case $d = 1$, there is a lower bound showing that testing monotonicity (for some constant ϵ) indeed requires $\Omega(\log n)$ queries [15]. In the high dimensional case, d is considered as the main parameter (and n might be as low as 2); a successful algorithm is typically one that is polynomial in $1/\epsilon$ and d. This case was first considered by Goldreich et al. [22] that showed an algorithm for testing monotonicity of functions over the boolean ($n = 2$) d-dimensional hyper-cube to a boolean range using $O(\frac{d}{\epsilon})$ queries. This result was generalized in [13] to arbitrary values of n, showing that $O(\frac{d \cdot \log^2 n}{\epsilon})$ queries suffice for testing monotonicity of general functions over $[n]^d$, which is the best known result so far.

1.1 Our Contributions

Our contributions are distribution-free testers for the two properties mentioned above: low-degree multivariate polynomials and low-dimensional monotone functions. We observe that the approach that stands behind the low-degree test can also be applied to the testing of other properties such as dictatorship and juntas functions [17, 32]. These algorithms are the first known distribution-free testers

[2] In the case $d = 1$ this yields a linear order.

for non-trivial properties. By this, we answer a natural question that has already been raised explicitly by Fischer [16, Subsection 9.3] and is implicit in [23]. We emphasize that our algorithms work for any distribution D without having any information about D.

Distribution-free low-degree testing for polynomials (and more). We show how to generalize the tester presented in [35] to a distribution-free tester with the same (up to a multiplicative constant factor of 2) query complexity $(O(d^2 + d \cdot \epsilon^{-1}))$. The algorithm and its analysis are presented in Section 3.

The generalization of the uniform tester to a distribution-free one is done, in this case, by adding another stage to the uniform tester. In this new stage, after verifying that the input function f is close to some low-degree polynomial g with respect to the uniform distribution, we check that f is also close to this specific polynomial g with respect to the given distribution D. For this purpose, our approach requires that we will be able to calculate the values of g efficiently based on the values on f. This is a generalization of the notion of self-correctors for single functions (see [10]) to classes of functions (which was previously introduced in [35]). We observe that the same approach can be used for distribution-free testing of every property that is testable in the uniform distribution and has a self-corrector in the above sense. The full details of this generalization appear in Section 4.

Distribution-free monotonicity testing. We present a distribution-free monotonicity tester in the low-dimensional hyper-cube case. Specifically, we present an algorithm whose complexity is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$ queries. This is done by first considering the one-dimensional case (the "line"). In this case, we prove that an algorithm of [14] can be slightly modified to deal with the distribution-free case with the same query complexity of $O(\frac{\log n}{\epsilon})$. Though it is possible to modify the original analysis for the distribution-free case, we choose to present a whole different analysis. We then show how to appropriately generalize this algorithm to deal with higher (yet, low) dimensions (a similar generalization approach was used in [9] for the uniform distribution case). The tester for the one-dimensional case and its generalization for higher dimensions appear in Section 5. Finally, we remark that it can be shown that distribution-free testing of monotonicity in the high-dimensional case cannot be done efficiently [11].

It is typical for known property testers to be quite simple and the analysis of why these algorithms work is where the property $\mathcal{P}$ in question requires understanding; indeed, Goldreich and Trevisan [25] proved that in certain settings this is an inherent phenomena: they essentially showed (with respect to the uniform distribution) that any graph-theoretic property that can be tested can also be tested (with a small penalty in the complexity) by a "generic" algorithm that samples a random subgraph and decides whether it has some property. Our work is no different in this aspect: our algorithms are similar to previously known algorithms and the main contribution is their analysis; in particular, that for the distribution-free case. Moreover, it is somewhat surprising that our distribution-free testers require no dramatically-different techniques than those used in the

construction and the analysis of previous algorithms (that work for the uniform distribution case). We remark, however, that although all the distribution-free testers presented in this work can be viewed as variations of testers for the uniform distribution, the modifications of the uniform-distribution testers in the various problems are different.[3]

2 Definitions

In this section, we formally define the notion of being ϵ-far from a property $\mathcal{P}$ with respect to a given distribution D defined over $\mathcal{X}$, and of distribution-free testing. Assume that the range of the functions in question is $\mathcal{A}$.

Definition 1. *Let D and $\mathcal{X}$ be as above. The* D-distance *between functions $f, g : \mathcal{X} \to \mathcal{A}$ is defined by $dist_D(f,g) \stackrel{def}{=} \Pr_{x \sim D}\{f(x) \neq g(x)\}$.*
The D-distance *of a function f from a property $\mathcal{P}$ (i.e., the class of functions satisfying the property $\mathcal{P}$) is $dist_D(f,\mathcal{P}) \stackrel{def}{=} \min_{g \in \mathcal{P}} dist_D(f,g)$.*
We say that f is (ϵ, D)-far from a property $\mathcal{P}$ *if $dist_D(f,\mathcal{P}) \geq \epsilon$.*

When the distribution in question is the uniform distribution over $\mathcal{X}$, we either use U instead of D or (if clear from the context) we omit any reference to the distribution.

Next, we define the notion of distribution-free tester for a given property $\mathcal{P}$.

Definition 2. *A* distribution-free tester *for a property $\mathcal{P}$ is a probabilistic oracle machine M, which is given a distance parameter $\epsilon > 0$, and an oracle access to an arbitrary function $f : \mathcal{X} \to \mathcal{A}$ and to sampling of a fixed but unknown distribution D over $\mathcal{X}$, and satisfies the following two conditions:*
1. If f satisfies $\mathcal{P}$, then $\Pr\{M^{f,D} = \mathit{Accept}\} = 1$.
2. If f is (ϵ, D)-far from $\mathcal{P}$, then $\Pr\{M^{f,D} = \mathit{Accept}\} \leq \frac{1}{3}$.

We note that a more general definition of testers that allows two-sided errors (as discussed in the introduction) is not needed here; all our testers, like many previously known testers, have one-sided error and always accept any function that satisfies the property $\mathcal{P}$ in question.

The definition of a uniform distribution tester for a property $\mathcal{P}$ can be derived from the previous definition by omitting the sampling oracle (since the tester can sample in the uniform distribution by itself) and by measuring the distance with respect to the uniform distribution.

Notice that since the distribution D in question is arbitrary, it is possible that there are two different functions f and g such that $dist_D(f,g) = 0$. Specifically, it is possible that $f \notin \mathcal{P}$ and $g \in \mathcal{P}$. Since the notion of testing is meant to be a relaxation of the notion of decision problems, it is required that the algorithm accepts (with high probability) functions that satisfy $\mathcal{P}$, but may reject functions that have distance 0 from $\mathcal{P}$ (but do not satisfy $\mathcal{P}$). This definition

[3] Indeed, in light of [23], there can be no generic transformation of uniform-distribution testers into distribution-free ones.

of distribution-free testing was introduced in [23, Definition 2.1]. In addition, note that the algorithm is allowed to query the value of the input function also in points with probability 0 (which is also the case with membership queries in learning theory)[4].

3 Distribution-free Low-Degree Testers for Polynomials

The first problem studied in the field of property testing was that of testing of multivariate polynomials (see [3, 6, 7, 10, 20, 35]). Let F be a finite field. In the problem of low-degree testing, with respect to the uniform distribution, the tester is given access to a function $f : F^m \to F$, a distance parameter ϵ, and a degree d, and has to decide whether f is a multivariate polynomial of total degree d, or is at least ϵ-far (with respect to the uniform distribution) from any degree d multivariate polynomial (i.e., one has to change the values of at least $\epsilon \times |F|^m$ points in order to transform f into a degree d multivariate polynomial; this implies that, for every degree d multivariate polynomial g, the probability that a *uniformly* drawn point x has a value $g(x)$ different than $f(x)$, is at least ϵ). Rubinfeld and Sudan ([35]) presented a tester for this problem with query complexity $O(d^2 + d \cdot \epsilon^{-1})$. We show how to modify this tester to a distribution-free tester with the same query complexity (up to a constant factor of 2).

3.1 Preliminaries

Fix some value for d and assume from now on that $|F| > 10d$. To describe the testers (both the one for the uniform distribution and our distribution-free one), we use the following terminology, from [35]:

A *line* in F^m is a set of $10d+1$ points of the form $\{x, x+h, \ldots, x+10dh\}$ for some $x, h \in F^m$. The line defined by x and h is denote $\ell_{x,h}$.

We say that a line $\ell_{x,h}$ *is an f-polynomial*, if there exists a univariate polynomial $P_{x,h}(i)$ of degree d, such that $f(x+ih) = P_{x,h}(i)$, for every $0 \leq i \leq 10d$.

Notice that if f is a multivariate polynomial of total degree at most d, then for every x and h, the line $\ell_{x,h}$ is an f-polynomial[5]. Given the values of f on a line $\ell_{x,h}$, testing whether this line is an f-polynomial can be done as follows:

- find, using interpolation, a univariate polynomial $P(i)$ of degree d, consistent with the values of f at the $d+1$ points $x, x+h, \ldots, x+dh$ (i.e., $P(i) = f(x+hi)$ for every $0 \leq i \leq d$).

[4] It is not known whether MQ are essential in general for testing even in the uniform case (see [33]); this is known only for specific problems such as monotonicity testing (see [14]).

[5] To see that, assume $f(x) = \sum_j a_j \prod_{l=1}^{d_j} x_{k_l^j}$, where a_j is the coefficient of the $j'th$ term in f, d_j is the degree ($d_j \leq d$), and k_l^j is the index of the $l'th$ variable in that term (note that $k_{l_1}^j$ is not necessarily different than $k_{l_2}^j$ for $l_1 \neq l_2$). In this case, for every fixed $x = (x_1, \ldots, x_m)$ and $h = (h_1, \ldots, h_m)$ the value $f(x+ih) = \sum_j a_j \prod_{l=1}^{d_j} (x_{k_l^j} + ih_{k_l^j})$, which, of course, is a degree d univariate polynomial in i.

– check, for every $(d+1) \leq i \leq 10d$, that $f(x+ih) = P(i)$. If so accept; otherwise reject.

We show how this basic test is used to build a uniform and a distribution-free low-degree test.

3.2 Low-degree test for the uniform distribution

The low-degree test for the uniform distribution is done by randomly sampling $O(d+\epsilon^{-1})$ lines (i.e., by uniformly choosing $x, h \in F^m$), and checking that each of these lines is an f-polynomial. The correctness of this algorithm follows immediately from the following theorem ([35, Theorem 9]).

Theorem 1. *There exists a constant c_U such that for $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, if f is a function from F^m to F, such that all but at most δ fraction of the lines $\{\ell_{x,h} | x, h \in F^m\}$ are f-polynomials, then there exists a polynomial $g : F^m \to F$ of total degree at most d such that $dist_U(f,g) \leq (1+o(1))\delta$ (provided that $|F| > 10d$).*

3.3 Distribution-free low-degree tester

Denote the class of multivariate polynomials of total degree d by $\mathcal{P}^d_{deg}$. In this section we show that the tester described in the previous subsection can be modified into a distribution-free tester for low-degree multivariate polynomials. That is, we present an algorithm with query complexity $O(d^2 + d \cdot \epsilon^{-1})$ that, given a distance parameter ϵ, a degree parameter d, and access to random sampling of F^m according to D and to membership queries of a function $f : F^m \to F$, distinguishes, with probability at least $\frac{2}{3}$, between the case that f is in $\mathcal{P}^d_{deg}$, and the case that f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$.

The natural generalization of the uniform-distribution tester above for the distribution-free case would be to replace the sampling of the tested lines by sampling according to the distribution D; i.e. sample the $O(d+\epsilon^{-1})$ lines by choosing $x \sim D$ and $h \sim U$ and check that these lines are f-polynomials. However, we do not know whether this modification actually works. Instead, the algorithm we present consists of two stages – in the first stage we simply run the uniform distribution test as is, and check that the function f is ϵ-close to $\mathcal{P}^d_{deg}$ with respect to the uniform distribution; the second stage is the generalization suggested above. We prove that this combined strategy actually works.

$\underline{Poly(\epsilon, d)}$

Set $k \stackrel{\text{def}}{=} \max\{\epsilon^{-1}, c_U \cdot d\}$. Repeat $5k$ times:

- Choose $x, h \in_R F^m$. If the line $\ell_{x,h}$ is not an f-polynomial, **return FAIL**.
- Choose $x \in_D F^m$, $h \in_R F^m$. If the line $\ell_{x,h}$ is not an f-polynomial, **return FAIL**.

return PASS

Theorem 2. *Algorithm $Poly(\epsilon, d)$ is a distribution-free tester for $\mathcal{P}^d_{deg}$; its query complexity is $O(d^2 + d \cdot \epsilon^{-1})$.*

The correctness of the algorithm relies on the following lemma:

Lemma 1. *Let c_U be the constant as above. For every $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, if f is a function from F^m to F such that*

- $\Pr_{x,h \sim U}\{\ell_{x,h}$ *is not an f polynomial*$\} \leq \delta$, *and*
- $\Pr_{x \sim D, h \sim U}\{\ell_{x,h}$ *is not an f polynomial*$\} \leq \delta$,

then there exists a polynomial $g : F^m \to F$ of total degree at most d such that $dist_D(f, g) \leq \frac{\delta}{1-40\delta} = (1 + o(1))\delta$ (provided that $|F| > 10d$).

The proof of the above lemma is omitted for lack of space. The proof is similar to ones presented in [35] and will appear in the full version of the paper.

Proof. **of theorem 2**.
To prove that the algorithm is indeed a distribution-free tester for $\mathcal{P}^d_{deg}$, we prove the following two facts:

1. If f is in $\mathcal{P}^d_{deg}$, then the algorithm accepts f with probability 1.
2. If f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$, then the algorithm $Poly(\epsilon, d)$ rejects f with probability at least $\frac{2}{3}$.

As explained before, if f is indeed a multivariate polynomial of total degree d, then every line is an f-polynomial. Hence, it follows that such f is accepted by the tester with probability 1. Assume from now on that f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$: Notice that, by the definition of k, for $\epsilon' = \frac{1}{k}$, f is (ϵ', D)-far from $\mathcal{P}^d_{deg}$. Based on Lemma 1, either $\Pr_{x,h \sim U}\{\ell_{x,h}$ *is not an f polynomial*$\} > \frac{\epsilon'}{2+40\epsilon'}$, or $\Pr_{x \sim D, h \sim U}\{\ell_{x,h}$ *is not an f polynomial*$\} > \frac{\epsilon'}{2+40\epsilon'}$ (otherwise, it follows that there exists a degree d polynomial g such that $dist_D(f, g) \leq \frac{\epsilon'}{(2+40\epsilon') \cdot (1-40\frac{\epsilon'}{2+40\epsilon'})} = \frac{\epsilon'}{2} < \epsilon'$, contradicting the fact that the D-distance of f from any such polynomial is at least ϵ'). Assume that the first event occurs. Therefore, the probability that a randomly chosen line $\ell_{x,h}$ is an f-polynomial is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})$. Hence, the probability that the algorithm accepts f is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})^{5k} = (1 - \frac{1}{2k+40})^{5k} \leq \frac{1}{e}^2 \leq \frac{1}{3}$ (the first inequality follows since $c_U \geq 100$ [35] implying that $k \geq 100$). Similarly, if the second event occurs, the probability that a randomly chosen line $\ell_{x,h}$, where $x \sim D$ and $h \sim U$, is an f-polynomial is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})$. Hence, as before, the probability that the algorithm accepts f is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})^{5k} \leq \frac{1}{3}$. □

4 Distribution-Free Testing of Properties with Self-Corrector

A careful examination and manipulation of the distribution-free tester presented in the previous section shows that, in fact, the only two features of low-degree multivariate polynomials used in the construction are:

- the existence of a one-sided error uniform distribution tester for low-degree polynomials, and
- the ability to efficiently compute (with high probability), in every point x of the domain, the correct value of the polynomial g that is close to the input function f, if f is indeed close to a multivariate low-degree polynomial. We refer to this ability as "property self-correction".

We argue that it is possible to construct a distribution-free tester for every property $\mathcal{P}$ that satisfies these two conditions. We first define the notion of a "property self-correction" formally (it has already been defined implicitly and used in [35]), and then introduce a general scheme for obtaining distribution-free testers for a variety of properties that satisfy the conditions.

The notion of "property self-corrector" is a generalization of the notion of self-correctors for functions introduced by Blum, Luby and Rubinfeld in [10]. A self-corrector for a *specific function* f is a randomized algorithm that given oracle access to a function g which is ϵ-close to f, is able to compute the value of f in every point of the domain. This definition can be generalized to classes of functions, specifically demanding that all the functions in the class are self-correctable using the same algorithm.

Definition 3. *An* ϵ self-corrector *for a property* $\mathcal{P}$ *is a probabilistic oracle machine* M*, which is given an oracle access to an arbitrary function* $f : \mathcal{X} \to \mathcal{A}$ *and satisfies the following condition:*
If there exists a function $g \in \mathcal{P}$ *such that* $dist_U(f, g) \leq \epsilon$ *(i.e.,* f *is* ϵ*-close to* $\mathcal{P}$*), then* $\Pr\{M^f(x) = g(x)\} \geq \frac{2}{3}$*, for every* $x \in \mathcal{X}$*. If* $f \in \mathcal{P}$*, then* $\Pr\{M^f(x) = f(x)\} = 1$ *for every* $x \in \mathcal{X}$*.*

Note that the definition of "property self-corrector" refers to distance measured only with respect to the uniform distribution, however, we still use these correctors for the construction of distribution-free testers. Observe that a necessary condition for the existence of an ϵ-self-corrector for a property $\mathcal{P}$ is that for every function f such that $dist_U(f, \mathcal{P}) \leq \epsilon$ (i.e., f is ϵ-close to $\mathcal{P}$ with respect to the uniform distribution), there exists a *unique* function $g \in \mathcal{P}$ that is ϵ-close to $\mathcal{P}$ (implying that ϵ cannot be too large). Notice that the property of monotonicity does not fulfill this requirement[6]. Hence, the distribution-free monotonicity tester that is presented in the next section requires a different approach.

[6] Consider for example the following function $f : [n] \to \{0, 1\}$: for every $1 \leq i \leq \frac{n}{2}$ set $f(i) = 1$, and for every $\frac{n}{2} + 1 \leq i \leq n$ set $f(i) = 0$. f is $\frac{1}{2}$-far from monotone, and it is $\frac{1}{2}$-close to both constant functions: 0 and 1.

Next, we describe the generalized distribution-free testing scheme. Let $\mathcal{P}$ be a property, let $T_{\mathcal{P}}$ be a uniform distribution tester for $\mathcal{P}$ with query complexity Q_T that has one-sided error, and let $C_{\mathcal{P}}$ be an ϵ' property self-corrector for $\mathcal{P}$ with query complexity Q_C. Let $\epsilon \leq \epsilon'$, and $f : \mathcal{X} \to A$.

Tester$_D(\epsilon)$
Run $T_{\mathcal{P}}^f(\epsilon)$. If $T_{\mathcal{P}}^f(\epsilon) =$ FAIL, then **return FAIL**
Repeat $\frac{2}{\epsilon}$ times:
    Choose $x \in_D \mathcal{X}$.
    Repeat twice: Run $C_{\mathcal{P}}^f(x)$; If $f(x) \neq C_{\mathcal{P}}^f(x)$, then **return FAIL** .
return PASS

Theorem 3. *Algorithm Tester$_D(\epsilon)$ is a distribution-free tester for $\mathcal{P}$ with query complexity $Q_T(\epsilon) + \frac{2}{\epsilon} \cdot Q_C$.*

Proof. It is obvious that the query complexity of the algorithm Tester$_D(\epsilon)$ is indeed as required. Hence, we only have to prove the correctness of the algorithm. To do so, we prove the following two facts:

- if $f \in \mathcal{P}$ then f is accepted by the algorithm with probability 1.
- if f is (ϵ, D)-far from $\mathcal{P}$, then f is rejected by Tester$_D(\epsilon)$ with probability at least $\frac{2}{3}$.

If f is indeed in $\mathcal{P}$, then it passes the uniform test with probability 1, and the value returned by the self-corrector is always identical to the value of f. Hence, it is clear that in this case f is accepted by the algorithm. Assume from now on that f is (ϵ, D)-far from $\mathcal{P}$. In this case we distinguish between two possibilities:

If f is (ϵ, U)-far from $\mathcal{P}$, then the probability that it passes the uniform test is at most $\frac{1}{3}$.

If f is (ϵ, U)-close to $\mathcal{P}$, then there exists a function $g \in \mathcal{P}$ such that $dist(f, g) \leq \epsilon$. However, since $dist_D(f, \mathcal{P}) \geq \epsilon$, we deduce that $dist_D(f, g) \geq \epsilon$ (in other words, $\Pr_{x \sim D}\{f(x) \neq g(x)\} \geq \epsilon$). If f is accepted by the algorithm then one of the two following events happened: either we failed to sample a point in which f and g differ, or we succeeded to sample such a point, but both runs of the self-corrector failed to compute the value of g in this point. The probability of the first event is at most $(1-\epsilon)^{\frac{2}{\epsilon}} \leq \frac{1}{e}^2 \leq \frac{1}{6}$, and by the definition of a property self-corrector the probability of the second event is at most $\frac{1}{3}^2 < \frac{1}{6}$. Therefore, the total probability that f is accepted by the algorithm is at most $\frac{1}{3}$.

Hence, in both cases the probability that f is accepted by the algorithm is at most $\frac{1}{3}$. □

Remark 1. We used the assumption that there exists a uniform distribution test for the property $\mathcal{P}$ that has one-sided error. However, the same transformation can be applied also when the uniform distribution tester has two-sided error, only that the resulting distribution-free tester as well has two-sided error.

As was previously stated, the algorithm that was explicitly presented in Section 3 can actually be described as an application of this scheme for the

class of low-degree multivariate polynomials. Hence, instead of fully describing the distribution-free tester and proving its correctness, it was enough to show that this property can be tested in the uniform distribution and that it can be self-corrected. This scheme, however, also implies the existence of distribution-free testers for other properties. Among these properties are low-degree multivariate polynomials over $GF(2)$, juntas and dictatorships functions. A function $f : \{0,1\}^n \to \{0,1\}$ is said to be a k-junta if there exists a subset of $\{x_1, \ldots, x_n\}$ of size k that determines the value of f (i.e., f is independent of the other variable). A special case of juntas are dictatorship functions, where a single variable determines the value of the function. These properties (and other related properties) have uniform distribution testers, as was shown in [3, 17, 32]. In addition, they are all subsets of the class of low-degree polynomials over $GF(2)$ which is self correctable (for example, k juntas are a special case of degree k multivariate polynomials), and thus are self correctable (see [3] and [10]). Therefore, we can apply the scheme described in this section to obtain distribution-free testers for these properties.

Remark 2. Notice that given two properties $\mathcal{P}$ and $\mathcal{P}'$ such that $\mathcal{P}' \subseteq \mathcal{P}$, the fact that $\mathcal{P}$ is testable in the uniform distribution does not imply that $\mathcal{P}'$ is thus testable (to see this observe, for example, that every property is a subset of the class of all functions that is clearly testable). However, the fact that $\mathcal{P}$ is self-correctable implies that $\mathcal{P}'$ is self-correctable (using the same correction algorithm).

5 Distribution-free Monotonicity Testing on the d-Dimensional Cube

In this section, we present testers for monotonicity over the d-dimensional hypercube with respect to an arbitrary distribution D. As before, we assume D to be fixed but unknown, and beside the ability to sample according to D we assume no knowledge of D. For simplicity, we begin our discussion with the case $d = 1$, and show that given access to random sampling according to D and to membership queries, there is a distribution-free tester for monotonicity over $[n]$, whose query complexity is $O(\frac{\log n}{\epsilon})$. This algorithm can be generalized to a distribution-free tester for monotonicity over the d-dimensional hyper-cube whose query complexity is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$.

We begin with a few notations and definitions. Denote by $[n]$ the set $\{1, \ldots, n\}$, and by $[n]^d$ the set of d-tuples over $[n]$. For every two points $\boldsymbol{i}$ and $\boldsymbol{j}$ in $[n]^d$ we say that $\boldsymbol{i} \leq \boldsymbol{j}$ if for every $1 \leq k \leq d$, $i_k \leq j_k$. Let $(A, <_A)$ be some linear order.

Definition 4. *We say that a function $f : [n]^d \to A$ is* monotone *if for every $\boldsymbol{i}$ and $\boldsymbol{j}$ if $\boldsymbol{i} \leq \boldsymbol{j}$ then $f(\boldsymbol{i}) \leq_A f(\boldsymbol{j})$.*

Definition 5. *Let $f : [n]^d \to A$ be a function. A pair $(\boldsymbol{i}, \boldsymbol{j})$ is said to be an* f-violation *if $\boldsymbol{i} < \boldsymbol{j}$ and $f(\boldsymbol{i}) >_A f(\boldsymbol{j})$.*

Let D be any distribution on $[n]^d$, and let S be a subset of $[n]^d$. Define $\Pr_D\{\boldsymbol{i}\} \stackrel{\text{def}}{=} \Pr_{X \sim D}\{X = \boldsymbol{i}\}$, and $\Pr_D\{S\} \stackrel{\text{def}}{=} \sum_{\boldsymbol{i} \in S} \Pr_D\{\boldsymbol{i}\}$.

5.1 Testing monotonicity for the line ($d = 1$)

In this section we consider the case $d = 1$. Our algorithm is a variant of the algorithm presented in [14] for testing monotonicity, with respect to the uniform distribution. However, the analysis presented here for this algorithm is quite different. The algorithm works in phases, in each phase a "center point" is selected according to the distribution D (in the original algorithm, the center point is selected uniformly), and the algorithm looks for a violation of the monotonicity with this center point. The search for a violation is done by randomly sampling in growing neighborhoods of the center point. In other words, in the case $d = 1$, the only change made in the original algorithm in order to adjust it to be distribution-free is that the choice of center points is made according to D. However, the search for violations remains unchanged. It is important to observe that, when dealing with an arbitrary distribution, there is no connection between the distance of the function from monotone (or the probability of the violation) and the number of pairs that form a violation of monotonicity[7]. Hence, the correctness of the algorithm for the uniform distribution (i.e., the fact that in a function that is far from monotone we find a violation of monotonicity with high probability) does not imply its correctness for the general case.

Algorithm-monotone-1-dim$_D(f, \epsilon)$:
repeat $\frac{2}{\epsilon}$ times
  choose $i \in_D [n]$
  for $k \leftarrow 0 \ldots \lceil \log i \rceil$ **do**
    repeat 8 times
      choose $a \in_R [2^k]$
      if $f(i-a) >_A f(i)$ **then return FAIL**
  for $k \leftarrow 0 \ldots \lceil \log(n-i) \rceil$ **do**
    repeat 8 times
      choose $a \in_R [2^k]$
      if $f(i) >_A f(i+a)$ **then return FAIL**
return PASS

Theorem 4. *Algorithm monotone-1-dim$_D$ is a distribution-free monotonicity tester over the line with query complexity $O(\frac{\log n}{\epsilon})$.*

To prove this theorem, we need the following definitions and lemmas.

Lemma 2. *Let $f : [n] \to A$ be a function, and let $S \subseteq [n]$ be a set. If for every f-violation (i, j) either $i \in S$ or $j \in S$, then there exists a monotone function f' that differs from f only on points in S.*

A similar claim was proved in [13]; proof omitted. An immediate conclusion of the above lemma is the following:

[7] Observe, for example, the function $f : [n] \to [n]$ such that for every $0 \leq i \leq n-2$ we set $f(i) = i$, $f(n-1) = n$ and $f(n) = n-1$. Set the distribution D to be $D(n-1) = D(n) = \frac{1}{2}$.

Lemma 3. *Let $f : [n] \to A$ be a function (ϵ, D)-far from monotone. Given $S \subseteq [n]$, if for every f-violation (i,j) either $i \in S$ or $j \in S$, then* $\Pr_D\{S\} \geq \epsilon$.

Definition 6. *For an f-violation (i,j), we say that i is* active *in this violation if*

$$|\{k \ : \ i < k < j \ , \ f(i) >_A f(k)\}| \ \geq \ \frac{j-i-1}{2},$$

similarly, j is active *in this violation if* $|\{k \ : \ i < k < j \ , \ f(j) <_A f(k)\}| \ \geq \ \frac{j-i-1}{2}$.

That is, i is active in an f-violation (i,j), if for at least half of the points $i < k < j$, (i,k) is also an f-violation (i.e., $f(i) >_A f(k)$).

Observation 1: For every f-violation (i,j), at least one of i and j is active in (i,j). (Proof omitted)

Define the active set of f (denoted A_f) as the set of all points that are active in some f-violation. Following this observation and applying Lemma 3 to the set A_f, if f is (ϵ, D)-far from monotone then $\Pr_D\{A_f\} \geq \epsilon$. We turn now to prove Theorem 4.

Proof. It is easy to see that the query complexity of the algorithm is as required. Hence, we are left to prove that it is indeed a distribution-free tester. The fact that every monotone function f is accepted by the algorithm follows immediately from its definition. From now on, assume that f is (ϵ, D)-far from monotone; we prove that f is rejected with probability at least $\frac{2}{3}$. Our algorithm may fail to detect that f is not monotone if either one of the following two events occurs:

1. None of the points sampled by the algorithm according to D is in A_f.

2. The algorithm picked at least one point $i \in A_f$, but failed to detect that i belongs to some f-violation.

It is easily verified that the probability of the first event is at most $(1-\epsilon)^{\frac{2}{\epsilon}} \leq \frac{1}{e^2} \leq 1/6$. We now turn to bound the probability of the second event. By the definition of A_f, for every $i \in A_f$ there is a j such that either (i,j) or (j,i) is an f-violation and i is active in this violation. Assume w.l.o.g. that (i,j) is an f-violation. For $k = \min\{l \ : \ 2^l \geq j - i\}$ (i.e., k is the smallest integer s.t. $j \leq i + 2^k$), we can claim that $|\{l \mid i < l \leq i + 2^k, f(i) >_A f(l)\}|$ is more than $\frac{1}{4} \cdot 2^k$. This is due to the fact that $j - i > 2^{k-1}$, and since i is active in the f-violation (i,j), for at least half the points l between i and j (i.e., at least $\frac{2^{k-1}}{2}$ points) the pair (i,l) is an f-violation. The probability that the algorithm fails to find an f-violation for this k is at most $(\frac{3}{4})^8 \leq \frac{1}{6}$, and hence the probability of the second event is at most $\frac{1}{6}$, implying that the total probability that the algorithm will wrongly accept f is at most $\frac{1}{3}$. □

Remark 3. In the journal version of [14], an additional tester for monotonicity on the line, called "Sort-Check-II", is presented. This algorithm can also be transformed to be a distribution-free monotonicity tester over the line. However, we do not know if it can be generalized to higher dimensions.

We saw how to test monotonicity over the one-dimensional hyper-cube (the line) when the distance is measured with respect to an arbitrary distribution. It is possible to generalize this algorithm to the d-dimensional case. The full details of the generalized algorithm and its analysis are omitted from this version and will appear in the full version of this paper.

References

1. N. Alon, S. Dar, M. Parnas, and D. Ron, *Testing of clustering. In Proceedings of the 41^{st} Annual Symposium on Foundations of Computer Science,* pages 240–251, 2000.
2. N. Alon, E. Fischer, M. Krivelevich, and M. szegedy, *Efficient testing of large graphs. In Proceedings of the 40^{th} Annual Symposium on Foundations of Computer Science,* pages 656–666, 1999.
3. N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron, *Testing low-degree polynomials over $GF(2)$. To appear in Proceedings of Random 2003.*
4. N. Alon, M. Krivelevich, I. Newman, and M. Szegedy, *Regular languages are testable with a constant number of queries, SIAM Journal on Computing* 30:1842–1862, 2001 (also appeared in Proceedings of the 40^{th} Annual Symposium on Foundations of Computer Science, pages 645–655, 1999).
5. N. Alon and A. Shapira, *Testing satisfiablity. In Proccedings of 13^{th}* SODA, 2001.
6. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and the hardness of approximation problems, JACM,* 45(1):501–555, 1998, (preliminary version appeared in Proc. 33^{th} FOCS, 1992).
7. S. Arora and S. Safra, *Probabilistic checkable proofs: A new characterization of NP. JACM,* 45(1):70–122, 1998 (a preliminary version appeared in Proc. 33^{rd} FOCS, 1992).
8. S. Arora and M. Sudan, *Improved low-degree testing and its applications. Proceedings of the 29^{th} ACM STOC 485–495,1997.*
9. T. Batu, R. Rubinfeld, and P. White, *Fast approximation PCPs for multidimensional bin-packing problems, Proceedings of the 3^{rd} International Workshop on Randomization and Approximation Techniques in Computer Science* 246–256,1999.
10. M. Blum, M. Luby, and R. Rubinfeld, *Self testing/correcting with applications to numerical problems, Journal of Computer and System Sceince* 47:549–595, 1993.
11. N. Bshouty, *Private communication.*
12. A. Czumaj and C. Sohler, *Testing hypergraph coloring, ICALP 2001,* 493–505.
13. Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky, *Improved testing algorithms for monotonicity, Proceedings of the 3^{rd} International Workshop on Randomized and Approximation Techniques in Computer Science,* pages 97–108, 1999.
14. E. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan, *Spot-checkers, Journal of Computing and System Science*, 60:717–751, 2000 (a preliminary version appeared in Proc. 30^{th} STOC ,1998).
15. E. Fischer, *On the strength of comparisons in property testing, manuscript (available at ECCC TR00-083).*
16. E. Fischer, *The art of uninformed decisions: A primer to property testing, The Computational Complexity Column of The bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.

17. E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky, *Testing Juntas, Proceedings of the* 43^{rd} *FOCS 103–112,2002.*
18. E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld and, A. Samorodnitsky, *Monotonicity testing over general poset domains, Proceedings of the* 34^{th} *ACM STOC 474–483,2002.*
19. E. Fischer and I. Newman, *Testing of matrix properties, Proceedings of the* 33^{rd} *ACM STOC,* pages 286–295, 2001.
20. P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self testing/correcting for polynomials and for approximate functions. In Proceedings of the* 23^{rd} *Annual ACM Symposium on Theory of Computing,* pages 32–42, 1991.
21. O. Goldreich, *Combinatorical property testing – a survey, In: Randomized Methods in Algorithms Design* (P. Pardalos, S. Rajasekaran and J. Rolim eds.), AMS-DIMACS pages 45–61, 1998 .
22. O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky, *Testing Monotonicity, Combinatorica*, 20(3):301–337, 2000 (a preliminary version appeared in Proc. 39^{th} FOCS, 1998).
23. O. Goldreich, S. Goldwasser, and D. Ron, *Propert testing and its connection to learning and approximation, Journal of the ACM*, 45(4):653–750, 1998 (a preliminary version appeared in Proc. 37^{th} FOCS, 1996).
24. O. Goldreich and D. Ron, *Property testing in bounded degree graphs. In Proccedings of the* 31^{st} *Annual ACM Symposium on the Theory of Computing,* pages 406–415, 1997.
25. O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties. In Proceedings of FOCS* 2001, pages 460–469.
26. M. J. Kearns and U. V. Vzirani, *An introduction to Computational Learning Theory*, MIT Press, 1994.
27. Y. Kohayakawa, B. Nagle, and V. Rodl, *Efficient testing of hypergraphs. In Proceedings of ICALP 2002.*
28. I. Newman, *Testing of functions that have small width branching programs. In Proceedings of the* 41^{st} *Annual Symposium of Foundations of Computer Science,* pages 251–258, 2000.
29. M. Parnas, and D. Ron, *Testing metric properties. In Proceedings of the* 33^{rd} *ACM STOC (2001)*, pages 276–285.
30. M. Parnas, and D. Ron, *Testing the diameter of graphs, RANDOM APPROX (1999), 85–96.*
31. M. Parnas, D. ron, and R. Rubinfeld, *Testing parenthesis languages. In Proceedings of the* 5^{th} *International Workshop on Randomization and Approximation Techniques in Computer Science (2001)*, pages 261–272.
32. M. Parnas, D. Ron, and A. Samorodnitsky, *Proclaiming dictators and juntas ot testing boolean formulae, RANDOM APPROX (2001), 273–284.*
33. D. Ron, *Property testing (a tutorial), In: Handbook of Randomized Computing* (S.Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolin eds), Kluwer Press (2001).
34. R. Rubinfeld, *Robust functional equations and their applications to program testing. In: SIAM Journal on Computing,* 28(6):1972–1997, 1999 (appeared in Proceedings of the 35^{th} Annual Symposium of Foundations of Computer Science, 1994).
35. R. Rubinfeld and M. Sudan, *Robust characterization of polynomials with applocations to program testing, SIAM Journal of Computing*, 25(2):252–271, 1996. (first appeared as a technical report, Cornell University, 1993).